CS436 Cloud Computing Term Project

Spring 2023-2024

Project Members

Mehmet Enes Onuş 29353

Akif Işıtan 29354

Bilal Berkam Dertli 29267

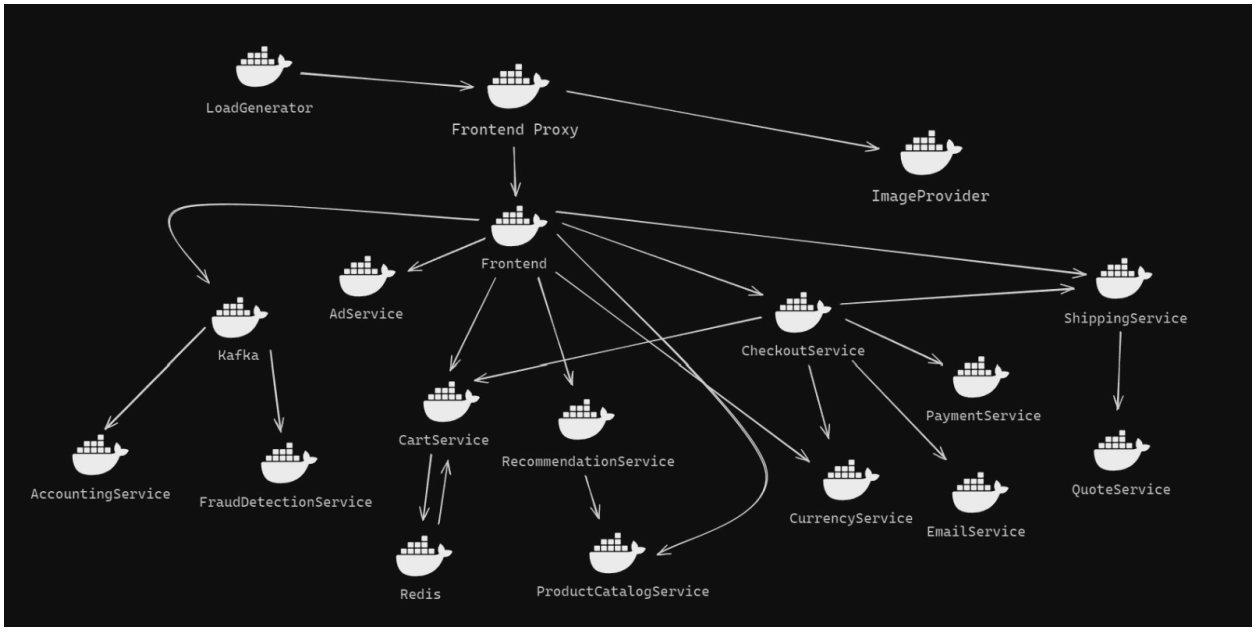
Github Repository

https://github.com/enesonus/cs436-project-opentelemetry

Plan

As a team we decided to use Open Telemetry's demo project that consists of many services, each running as a container, thus giving us a very nicely constructed Cloud Native application while giving us advantage of a load generator tool, Locust and observability tools like Jaeger and Grafana. At the initial phase of this project we plan to run these services on our machine and try to understand the logic of the system, after this we will work on understanding the scaling and orchestration of the system. This demo project has very nice guides on how to deploy the services both using docker compose and Kubernetes. We will use these guides to understand the logic and running mechanism behind Cloud Native applications and explore the key points that make Cloud Computing useful. In addition to these since OpenTelemetry is an observability framework, this demo project also gives us the advantage of getting detailed information using logs, traces and metrics and visualizing them with Grafana and Jaeger.

Here is the architectural design graphic of the project we are using:



Proposed Cloud Architecture

Cloud Services
Utilize Google Cloud for hosting the microservices as containers using Google Cloud Run.

Service Deployment
Deploy microservices using Kubernetes for orchestration to manage and scale the services dynamically.
Use containerized environments for each service to maintain consistency and isolation.

Telemetry and Monitoring
Leverage the OpenTelemetry collector within the Kubernetes cluster to gather telemetry data from all microservices.
Configure exporters in the OpenTelemetry collector to send metrics and traces of all containers to Prometheus and Jaeger respectively.

High Availability
Set up multi-zone or multi-region deployment in the cloud to ensure high availability.

Scalability
Use auto-scaling features of Kubernetes to handle variations in load.
Leveraging Google Cloud’s auto scaling the system scaling containers.